UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/637,078	08/11/2000	Erik R Altman	YOR9-2000-0415US1 (8728-4	8733

46069 7590 11/17/2006

F. CHAU & ASSOCIATES, LLC
130 WOODBURY ROAD
WOODBURY, NY 11797

EXAMINER	
ART UNIT	PAPER NUMBER

DATE MAILED: 11/17/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/637,078
Filing Date: August 11, 2000
Appellant(s): ALTMAN ET AL.

Frank Chau
For Appellant

SUPPLEMENTAL EXAMINER'S ANSWER

This is in response to the appeal brief filed 17 April 2006 appealing from the Office action mailed 06 October 2005.

(1) Real Party in Interest

A statement identifying by name the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix to the brief is correct.

(8) Evidence Relied Upon

5,590,354	Klapproth et al.	12-1996
6,622,300	Krishnaswamy et al.	9-2003
5,355,484	Record et al.	10-1994

"Dictionary of Computing" Fourth Edition, Oxford University Press, 1996, pp. 26 and 432.

Chang et al. "Using Profile Information to Assist Classic Code Optimizations", Software Practice and Experience, Vol. 21(12), December 1991, pp. 1301-21.

Altman et al. "DAISY: Dynamic Compilation for 100% Architectural Compatibility", Proceedings of the 24th Annual International Symposium on Computer Architecture (ISCA 97), Denver, CO, ACM, June 1997, pp. 26-37.

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 102/103

1. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this

> subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.
>
> (e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraphs (1), (2), and (4) of section 371(c) of this title before the invention thereof by the applicant for patent.

The changes made to 35 U.S.C. 102(e) by the American Inventors Protection Act of 1999 (AIPA) and the Intellectual Property and High Technology Technical Amendments Act of 2002 do not apply when the reference is a U.S. patent resulting directly or indirectly from an international application filed before November 29, 2000. Therefore, the prior art date of the reference is determined under 35 U.S.C. 102(e) prior to the amendment by the AIPA (pre-AIPA 35 U.S.C. 102(e)).

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made

3. Claims 1, 4-8, 11-13, 16-17, 22, 38 are rejected under 35 U.S.C. 102(e) as anticipated by **Krishnaswamy** et al. (USPN 6,622,300) or, in the alternative, under 35 U.S.C. 103(a) as obvious over **Krishnaswamy** et al. (USPN 6,622,300) in view of **Klapproth** et al. (USPN 5,590,354).

Claim 1

Krishnaswamy's background disclosed a method for profiling computer program executions in a computer processing system having a processor and a memory hierarchy *(column 1, lines 10-43)*, comprising the steps of:

- executing a computer program *(column 1, lines 33-35)*;
- storing, in a memory array, profile counts for a plurality of events associated with the execution of the computer program *(column 1, lines 36-39)*
- selecting at least one of the plurality of events for profiling *(column 6, lines 24-28, at some point in time the events to be profiled are selected)*;
- updating the profile counts for only the events *(column 1, lines 33-37)*; and
- assisting compilation and optimization of the computer program, based upon the profile counts stored in the memory array *(column 1, lines 33-43)*.

Depending on interpretation of "the memory array being separate and distinct from the memory hierarchy", **Krishnaswamy** is anticipated the claimed invention through column 5, lines 35-45, column 6, lines 34-36 and figure 2, elements 90, 70, 100, 110, 160 and 170. The passages and figure illustrate a separate and distinct memory that is not perturbed by the profile. At least one interpretation is the profile information being stored in the elements 90 being separate and distinct from elements 70, 170 and 160. Alternatively, if the profile data is stored in the kernel space then the shared user memory is considered the separate memory or at very least the separate and distinct

memory is the removable media unit 170 and the permanent storage 160. This is the broadest reasonable interpretation.

Otherwise, **Klapproth** demonstrated that it was known at the time of invention to provide a separate and distinct memory for tracing and profiling (column 2, lines 3-40, figure 1, element 58, column 3, lines 16-21). It would have been obvious to one of ordinary skill in the art at the time of invention to implement trace/profiling system of **Krishnaswamy** with a separate buffer for storing as found in **Klapproth**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide an interface that doesn't unduly burden the bus (column 2, lines 32-33).

Claim 4

Krishnaswamy disclosed the method according to claim 1, wherein said updating step is triggered by execution of the events *(column 6, lines 21-33).*

Claim 5

Krishnaswamy did not explicitly state the method according to claim 1, wherein said updating step is triggered by execution of instructions embedded into an instruction stream of the computer program. **Krishnaswamy** demonstrated that it was known at the time of invention to instrument code for profiling (column 1, lines 56-57). It would have been obvious to one of ordinary

skill in the art at the time of invention to implement the profiling-based optimizing compiler of **Krishnaswamy** with instrumented code as found in **Krishnaswamy**'s own teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to allow for collection of a minimum amount of data, thus saving space and time (column 1, lines 60-62), additionally not all processors are equipped with performance monitoring functions and thus instrumentation is required for profiling.

Claim 6

Krishnaswamy disclosed the method according to claim 1, further comprising the step of detecting whether a profile count has exceeded an adjustable predefined threshold *(column 6, lines 30-34).*

Claim 7

Krishnaswamy disclosed the method according to claim 1, further comprising the step of indicating when a profile count has exceeded an adjustable predefined threshold *(column 6, lines 30-34).*

Claim 8

Krishnaswamy disclosed the method according to claim 7, wherein said indicating step comprises the step of raising an exception *(column 6, lines 30-34).*

Claim 11

Krishnaswamy disclosed the method according to claim 1, further comprising the step of identifying profile information corresponding to the profile counts using a profiling event identifier *(column 6, lines 26-36; column 1, lines 34-43; identification of some sort required for proper usage of collected information).*

Claim 12

Krishnaswamy disclosed the method according to claim 11, further comprising the step of addressing the memory array, using the profiling event identifier *(column 6, lines 24-36; column 1, lines 34-43; identification of some sort required for proper usage of collected information).*

Claim 13

Krishnaswamy disclosed the method according to claim 1, further comprising the steps of: generating the profile counts using profile counters associated with the events *(column 6, lines 24-28).* **Krishnaswamy** did not explicitly state maintaining the profile counters in a set-associate manner. Official Notice is taken that it was known at the time of invention to store values in a set-associative manner. It would have been obvious to one of ordinary skill in the art at the time of invention to implement the memory of **Krishnaswamy** with a set associative manner. This implementation would have been obvious

because one of ordinary skill in the art would be motivated to make use of a regular method of memory, which thus common and easy to use/implement.

Claim 16

Krishnaswamy disclosed the method according to claim 1, further comprising the step of supporting read operations from the memory array in an off-line optimization of the program *(column 1, lines 30-43)*.

Claim 17

Krishnaswamy disclosed the method according to claim 1, further comprising the step of assisting optimization of the program, based upon the profile counts stored in the memory array *(column 1, lines 34-37)*.

Claim 22

Krishnaswamy disclosed the method according to claim 1, wherein the memory hierarchy includes data and instruction caches, and the memory array is separate and distinct from the memory hierarchy so as to not perturb normal operations of the data and instruction caches *(Figure 2; and as above for claim 1)*.

Claim 38

Krishnaswamy disclosed the method according to claim 1, wherein said method is implemented by a program storage device readable by machine, tangibly embodying a program of instructions executable by the machine to perform said method steps *(column 1, lines 34-37; compiler).*

4. Claims 3, 9-10, 23-30, 32-34, 37 and 39 rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishnaswamy** et al. (USPN 6,622,300) in view of "Dictionary of **Computing**".

Claim 3

Krishnaswamy did not explicitly state the method according to claim 1, wherein said storing and updating steps are performed asynchronously to prevent a decrease of an execution speed of the computer program. **Computing** demonstrated that it was known at the time of invention to perform circuit operations asynchronously (page 26, asynchronous circuit). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the system of **Krishnaswamy** with an asynchronous circuit design, including storing and updating counts as suggested by **Computing**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to allow operation with a minimum of delay (page 26, asynchronous circuit).

Claim 9

Krishnaswamy disclosed the method according to claim 1, further comprising the steps of: accumulating profile updates (**Krishnaswamy**: column 1, lines 34-37). **Krishnaswamy** did not explicitly state dividing the accumulated profile updates by a threshold fraction. **Computing** demonstrated that it was known at the time of invention to make use of scaling (page 432). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the profiling counters of **Krishnaswamy** with scaling (or dividing/multiplying by a threshold fraction) the update value as found in **Computing**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to adjust the stored value to the hardware/equipment (register size) limitations (**Computing**: page 432).

Claim 10

Krishnaswamy did not explicitly state disclosed the method according to claim 1, further comprising the step of scaling the profile counts to prevent profile information overflow. **Computing** demonstrated that it was known at the time of invention to make use of scaling (page 432). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the finite hardware memory of **Krishnaswamy** with scaling the update value as found in **Computing**'s teaching. This implementation would have been obvious because

one of ordinary skill in the art would be motivated to adjust the stored value to the hardware/equipment (register size) limitations (**Computing**: page 432).

Claim 23

The limitations of claim 23 correspond to the limitations of claims 1 and 10 and as such are rejected in the same manner.

Claims 24-30, 32-34, 37 and 39

The limitations of claims 24-30, 32-34, 37 and 39 correspond to the limitations of claims 3-9, 11-13, 22 and 17 and are dependent upon claim 23. Thus, the claims are rejected in the same manner as 3-9, 11-13, 22 and 17 in consideration of claim 23.

5. Claims 14-15 rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishnaswamy** et al. (USPN 6,622,300) in view of **Record** et al. (USPN 5,355,484).

Claims 14 and 15

Krishnaswamy did not explicitly state the method according to claim 13, further comprising the step of selecting a profile counter to be evicted from the memory array based upon a predefined replacement, when a number of profiling events assigned to an associative class of events is exceeded. **Record**

demonstrated that it was known at the time of invention to perform the above operation (column 9, lines 13-20). **Record** further demonstrated (as found in claim 15) that it was known at the time of invention to utilize the replacement strategy based upon on of least-recently-used and first-in-first-out (column 9, lines 13-20). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the optimizing profiling system of **Krishnaswamy** with the control provided by **Record**. This implementation would have been obvious because one of ordinary skill in the art would be motivated to minimize any reduction in execution time resulting from profiling a system by limiting the number of events to be monitored (**Record**: column 2, lines 17-25).

6. Claims 35 and 36 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishnaswamy** et al. (USPN 6,622,300) in view of Dictionary of **Computing**" in further view of **Record** et al. (USPN 5,355,484).

Claims 35-36

The limitations of claims 35 and 36 correspond to the limitations of claims 14 and 15 and are indirectly dependent upon claim 23. Thus, the claims are rejected in the same manner as 35 and 36 in consideration of claim 23.

7. Claims 18-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishnaswamy** et al. (USPN 6,622,300) in view of **Altman** et al., "DAISY: Dynamic Compilation for 100% Architectural Compatibility".

Claim 18

Krishnaswamy did not explicitly state the method according to claim 1, wherein said assisting step is performed during at least one of dynamic binary translation and dynamic optimization [compilation] of the computer program. **Altman** demonstrated that it was known at the time of invention to provide dynamic binary translation and dynamic optimization [compilation] (page 27, section 2 and page 28, section 2.1; additionally page 27, left column, last three paragraphs). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the profiling compiler system of **Krishnaswamy** with dynamic translation and optimization [compilation] as found in **Altman**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide compiling/translating system with dynamic operation (useful for providing real-time operation; page 27, left column, second and third paragraphs) and profiling for optimization (useful for helping code execute better).

Claim 19

Krishnaswamy and **Altman** disclosed the method according to claim 18, wherein the dynamic binary translation and dynamic optimization of the computer program results in translated and optimized code, respectively, the translated and optimized code comprising instructions groups which pass control there between *(**Krishnaswamy**: column 1, lines 30-43; and **Altman**: page 27, right column, third paragraph; page 29, section 3).*

8. Claims 20 and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishnaswamy** et al. (USPN 6,622,300) in view of **Altman** et al., "DAISY: Dynamic Compilation for 100% Architectural Compatibility" in further view of **Chang** et al., "Using Profile Information to Assist Classic Code Optimizations".

Claims 20 and 21

Krishnaswamy and **Altman** did not explicitly state the method according to claim 19, further comprising the step of identifying frequently executed paths of the computer program, by instrumenting exits from the instruction groups with a profiling instruction that indicates a unique group exit identifier. **Chang** demonstrated that it was known at the time of invention to instrument groups of instructions to provide an ID (page 1305-1306, item (a) under "Profiler implementation") and to optimize frequently executed paths (page 1306,

bottom). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the optimizing profiling compiler of **Krishnaswamy** and **Altman** with group instrumentation as found in **Chang**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to optimize frequently executed program paths (page 1301, Introduction). **Chang** did not explicitly state to instrument exit points. Official Notice is taken that it was known at the time of invention to instrument exits. Furthermore, **Chang** demonstrated instrumenting the entry point (page 1305-1306, item (a) under "Profiler implementation") or taken more generally simply ensuring instrumentation of the group/function. It would have been obvious to one of ordinary skill in the art at the time of invention to instrument exits of groups/functions in the compiler of **Krishnaswamy**, **Altman** and **Chang**. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide an information about the profiled code, which includes determining if a group/function is executed. Both entry and exit points are the most obvious instrumentation points of all locations, since the are easily identifiable. Additionally, **Krishnaswamy** and **Altman** did not explicitly state the method according to claim 19, further comprising the step of extending the instruction groups along a frequently executed path. However, **Chang** demonstrated this as well on page 1306, items (b) through (e) and page 1301-1302, "Introduction".

9. Claims 40 and 42 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishnaswamy** et al. (USPN 6,622,300) in view of **Chang** et al., "Using Profile Information to Assist Classic Code Optimizations".

Claim 40

Krishnaswamy disclosed a method for profiling computer program executions in

a computer processing system having a processor and a memory hierarchy, comprising the steps of:

- executing a computer program *(column 1, lines 33-35)*;
- storing, in a single memory array, a plurality of event-specific profile counts, each associated with an event associated with the execution of a path of the computer program *(column 1, lines 36-39)*
- selecting at least one of a plurality of event-specific profile counts for profiling the path of the computer program *(column 6, lines 24-28, at some point in time the events to be profiled are selected)*
- updating the profile counts for only the events *(column 1, lines 33-37)*

Depending on interpretation of "the memory array being separate and distinct from the memory hierarchy", **Krishnaswamy** may have anticipated the claimed invention through column 5, lines 35-45, column 6, lines 34-36 and figure 2,

elements 70, 100, 110, 160 and 170. The passages and figure illustrate a separate and distinct memory that is not perturbed by the profile. If the profile data is stored in the kernel space then the shared user memory is considered the separate memory or at very least the separate and distinct memory is the removable media unit 170 and the permanent storage 160. This is the broadest reasonable interpretation.

Krishnaswamy did not explicitly state if at least one of the selected event-specific counts has exceeded a predefined threshold, optimizing the portions of the computer program associated with the event-specific profile counts more aggressively than other portions of the computer program. **Chang** demonstrated that it was known at the time of invention to optimize more heavily various parts of a program based upon threshold (pages 1306-1308, "Optimizing frequently executed paths" section). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the optimizing profiling compiler of **Krishnaswamy** with group instrumentation as found in **Chang**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to optimize frequently executed program paths primarily since they are executed more (page 1301, Introduction and pages 1306-1308).

Claim 42

Krishnaswamy did not explicitly state the method according to claim 40, wherein the memory array is arranged as a two-way set associative array. Official Notice is taken that it was known at the time of invention to store values in a two way set-associative manner. It would have been obvious to one of ordinary skill in the art at the time of invention to implement the memory of **Krishnaswamy** with a set associative manner. This implementation would have been obvious because one of ordinary skill in the art would be motivated to make use of a regular method of memory, which thus common and easy to use/implement.

10. Claims 41 is rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishnaswamy** et al. (USPN 6,622,300) in view of **Chang** et al., "Using Profile Information to Assist Classic Code Optimizations" in further view of **Altman** et al., "DAISY: Dynamic Compilation for 100% Architectural Compatibility".

Claim 41

Krishnaswamy did not explicitly state the method according to claim 40, further comprising the step of optimizing the computer program during at least one of static and dynamic compilation using the profile information. **Altman** demonstrated that it was known at the time of invention to provide dynamic binary translation and dynamic optimization [compilation] (page 27, section 2

and page 28, section 2.1; additionally page 27, left column, last three paragraphs). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the profiling compiler system of **Krishnaswamy** with dynamic translation and optimization [compilation] as found in **Altman**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide compiling/translating system with dynamic operation (useful for providing real-time operation; page 27, left column, second and third paragraphs) and profiling for optimization (useful for helping code execute better).

11. Claims 3, 9-10, 23-30, 32-34, 37 and 39 rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishnaswamy** et al. (USPN 6,622,300) in view of **Klapproth** et al. (USPN 5,590,354) in view of "Dictionary of **Computing**". Rejection of above not repeated here.

12. Claims 14-15 rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishnaswamy** et al. (USPN 6,622,300) in view of **Klapproth** et al. (USPN 5,590,354) in view of **Record** et al. (USPN 5,355,484). Rejection of above not repeated here.

13. Claims 35 and 36 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishnaswamy** et al. (USPN 6,622,300) in view of **Klapproth** et al. (USPN 5,590,354) in view of Dictionary of **Computing**" in further view of **Record** et al. (USPN 5,355,484). Rejection of above not repeated here.

14. Claims 18-19 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishnaswamy** et al. (USPN 6,622,300) in view of **Klapproth** et al. (USPN 5,590,354) in view of **Altman** et al., "DAISY: Dynamic Compilation for 100% Architectural Compatibility". Rejection of above not repeated here.

15. Claims 20 and 21 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishnaswamy** et al. (USPN 6,622,300) in view of **Klapproth** et al. (USPN 5,590,354) in view of **Altman** et al., "DAISY: Dynamic Compilation for 100% Architectural Compatibility" in further view of **Chang** et al., "Using Profile Information to Assist Classic Code Optimizations". Rejection of above not repeated here.

16. Claims 40 and 42 are rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishnaswamy** et al. (USPN 6,622,300) in view of **Klapproth** et al. (USPN 5,590,354) in view of **Chang** et al., "Using Profile Information to Assist Classic Code Optimizations". Rejection of above not repeated here.

Claim 40

Krishnaswamy disclosed a method for profiling computer program executions in

a computer processing system having a processor and a memory hierarchy, comprising the steps of:

- executing a computer program *(column 1, lines 33-35)*;
- storing, in a single memory array, a plurality of event-specific profile counts, each associated with an event associated with the execution of a path of the computer program *(column 1, lines 36-39)*
- selecting at least one of a plurality of event-specific profile counts for profiling the path of the computer program *(column 6, lines 24-28, at some point in time the events to be profiled are selected)*
- updating the profile counts for only the events *(column 1, lines 33-37)*

Klapproth demonstrated that it was known at the time of invention to provide a separate and distinct memory for tracing and profiling (column 2, lines 3-40, figure 1, element 58, column 3, lines 16-21). It would have been obvious to one of ordinary skill in the art at the time of invention to implement trace/profiling system of **Krishnaswamy** with a separate buffer for storing as found in **Klapproth**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide an interface that doesn't unduly burden the bus (column 2, lines 32-33).

Krishnaswamy did not explicitly state if at least one of the selected event-specific counts has exceeded a predefined threshold, optimizing the portions of the computer program associated with the event-specific profile counts more aggressively than other portions of the computer program. **Chang** demonstrated that it was known at the time of invention to optimize more heavily various parts of a program based upon threshold (pages 1306-1308, "Optimizing frequently executed paths" section). It would have been obvious to one of ordinary skill in the art at the time of invention to implement the optimizing profiling compiler of **Krishnaswamy** with group instrumentation as found in **Chang**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to optimize frequently executed program paths primarily since they are executed more (page 1301, Introduction and pages 1306-1308).

17. Claims 41 is rejected under 35 U.S.C. 103(a) as being unpatentable over **Krishnaswamy** et al. (USPN 6,622,300) in view of **Klapproth** et al. (USPN 5,590,354) in view of **Chang** et al., "Using Profile Information to Assist Classic Code Optimizations" in further view of **Altman** et al., "DAISY: Dynamic Compilation for 100% Architectural Compatibility". Rejection of above not repeated here.

(10) Response to Argument

Appellant's arguments filed 17 April 2006 have been fully considered but they are not persuasive. Appellant argues, within the Brief of 17 April 2006: A(i) **Krishnaswamy** fails to disclose "selecting at least one of the plurality of events for profiling" (page 8, first paragraph); A(ii) **Krishnaswamy** fails to disclose update counts for only the selected events and assisting compilation based upon the selected counts stored in the memory array (page 8, last sentence); A(iii) **Krishnaswamy** taken alone fails to disclose "storing, in a memory array, profile counts ... the memory array being separate and distinct from the memory hierarchy" (pages 9-10); B(i) the combination of **Krishnaswamy** and **Klapproth** fails to disclose profile counts stored separate from memory hierarchy (pages 12-13); C(iii) no disclosure of "a scaling circuit adapted to scale the profile counts to prevent profile information overflow" as found in claim 23 (page 15); and E(iii) no motivation to combine **Krishnaswamy** and **Chang** (page 19). Theses arguments are unpersuasive as will become clear in the below response. For convenience and clarity, the response is laid out according to Appellant's numbering scheme.

A(i)

First in response to Appellant's A(i) arguments, **Krishnaswamy** under the broadest reasonable interpretation does disclose "selecting at least one of the plurality of events for profiling" (column 6, lines 24-28). Specifically,

Krishnaswamy states, "counters programmable to count events like ...". At some point in time a counter is programmed to count an event and as such an event is selected to be profiled (this is not a statement that the counter will be *programmed to select*, but instead a statement where the act of programming is the selecting). Appellant even initially agrees, stating "the counters for **Krishnaswamy** can be programmed to count events A, B and C (17 April 2006 Brief: page 8, lines 4-5) and thus selecting events A, B and C for profiling. Appellant's arguments then, inexplicably, require an additional selection to occur, stating "that counters can be programmed to count events A, B and C does not necessarily imply that only A can be selected" (17 April 2006 Brief: page 8, lines 6-7). Claim 1 simply requires, "selecting at least one of the plurality of events for profiling". There is no requirement to make multiple selections or sub-selections of an original selection of events. This is the broadest reasonable interpretation. Additionally **Krishnaswamy** indicates other means of event selection, "[t]he dynamic optimization helper 230 then preferably uses that reported information to change the profile data that it collects through the PMUs 90 and to make decisions" (column 9, lines 63-66). Therefore, there is **Krishnaswamy** clearly discloses "selecting at least a plurality of events for profiling".

A(ii)

Second Appellant's argument A(ii) is necessarily overcome by the response immediately preceding. However, further analysis might prove

illuminating. Claim 1 requires three interrelated limitations: "storing ... counts for a plurality of events" (**Krishnaswamy** stores counts for the programmed/selected events), "selecting at least one of the plurality of events for profiling" (all the programmed/selected events previously mentioned as being stored), and "updating the profile counts for only the selected events" (as more counting occurs for all the programmed/selected events they too are stored).

A(iii)

Third in response to Appellant's A(iii), **Krishnaswamy** under the broadest reasonable interpretation does disclose "storing, in a memory array, profile counts ... the memory array being separate and distinct from the memory hierarchy" (column 5, lines 35-45, column 6, lines 34-36 and figure 2, elements 70, 100, 110, 160 and 170; at least one interpretation is the profile information being stored in the elements 90 being separate and distinct from elements 70, 170 and 160). Clearly, "memory hierarchy" as found in the claim language is a rather broad term with little limiting power. Appellant's own disclosure contextually defines the phrase as, "[t]he profile matrix 100 is a memory structure for storing profile information separate from the main memory hierarchy of the processor(s)" (emphasis added; Specification: page 14, lines 8-10). Referring to figure 5, Cache memory 506 and main memory 508 can clearly be seen as well as a profile matrix 100. The corresponding Specification sections state, "[t]he profile matrix 100 is a separate hardware

unit, which is distinct from the memory hierarchy" (page 22, lines 17-18). Thus, memory hierarchy, under the broadest reasonable interpretation, is defined by Appellant to include the cache hierarchy and the main memory (the only memory elements other than the profile matrix). Main memory is defined as Random Access Memory (RAM), the main general purpose storage region to which a microprocessor has direct access (also known as Primary Storage; see Microsoft Computer Dictionary). Secondary Storage refers to a computer's disks and tapes and so forth. Yet, with no support from Appellant's originally filed disclosure, Appellant now asserts "memory hierarchy" includes, "registers, cache, main memory, secondary storage (disks), offline storage (tapes)" (Brief: page 10, first paragraph). In summary, Appellant's originally filed disclosure only identifies a cache hierarchy and main memory for inclusion in the memory hierarchy; **Krishnaswamy** demonstrates usage of secondary storage for storing the profile counts, not main memory or cache memory (and therefore separate and distinct from the memory hierarchy); and Appellant now asserts an external definition to compensate for the lacking original disclosure in order to overcome the properly cited prior art.

Now continuing with Appellant's broad definition of "memory hierarchy" found in the Brief, it becomes clear Appellant's cited definition would necessarily include the "profile matrix" as the definition includes registers and caches. Appellant's originally filed disclosure states, "[an] optimized profile matrix may consist of a hierarchy of profile matrices (e.g., similar to caching

hierarchies)" (Specification: page 19, lines 1-2). Yet, Appellant asserts some forms of memory are within the "memory hierarchy" and some forms of memory, the "profile matrix", are distinct from the "memory hierarchy" (see originally Specification and Brief). The only guidance, in order clear-up this discrepancy and provide a reasonable interpretation of the claim language, is the originally filed disclosure (figure 5). And as discussed above, figure 5 indicates the memory hierarchy be limited to "cache hierarchy" and "main memory". Under this, the broadest reasonable interpretation, it becomes clear **Krishnaswamy** is read upon by the claim language. **Krishnaswamy** does not indicate cache hierarchy or main memory (primary storage) being used, but instead explicitly states secondary storage is being used, "Kernel memory space".

Appellant's conveniently fluid use of the phrase "memory hierarchy" to weigh against properly cited prior art, but not against Appellant's own claims, will not suffice. Simply, Appellant has failed to narrowly define the phrase, other than by explicit example, in the originally filed disclosure of the claimed invention. Therefore, the limitless phrase is open to reasonable interpretation. In the particular instance at hand, the "memory hierarchy" is limited to the memory elements to which profile counts are not stored.

B(i)

Fourth in response to Appellant's B(i), **Krishnaswamy** and **Klapproth** do disclose a memory "separate and distinct" from the memory hierarchy for

storing counts (figure 1, element 58; column 3, lines 16-21; column 2, lines 3-40). Appellant asserts this "memory is quite restricted and was not designed to store profile counts or a plurality of events associated with the execution of a program" (17 April 2006 Brief: page 12, end of second paragraph). Of course the rejection stated,

> **Klapproth** demonstrated that it was known at the time of invention to provide a separate and distinct memory for tracing and profiling (column 2, lines 3-40, figure 1, element 58, column 3, lines 16-21). It would have been obvious to one of ordinary skill in the art at the time of invention to implement trace/profiling system of **Krishnaswamy** with a separate buffer for storing as found in **Klapproth**'s teaching. This implementation would have been obvious because one of ordinary skill in the art would be motivated to provide an interface that doesn't unduly burden the bus (column 2, lines 32-33).

Klapproth merely disclosed a memory for storing tracing/profiling data separate and distinct from the memory hierarchy and provided an obvious reason for so doing. One of ordinary skill in the art would use that distinct memory *in combination* with **Krishnaswamy**, which clearly discloses storing profile counts for a plurality of events. The specific information stored in **Klapproth** is unimportant to this combination. If the memory of **Klapproth** were to meet the deficiencies Appellant asserts are required, the reference alone would have been cited against the claimed invention.

Appellant then contends the combination is not obvious. Yet, this argument (Brief: page 13, first paragraph) ignores the stated reason for

obvious previously provided, "[t]his implementation would have been obvious because one of ordinary skill in the art would be motivated to provide an interface that doesn't unduly burden the bus (column 2, lines 32-33)". Appellant states, "[i]t is neither clear nor obvious why one would want ... an on-chip memory ... under so many layers and levels of circuit indirection with **Krishnaswamy**" (Brief: page 13, lines 5-8). Though Appellant has produced an asserted reason for non-obviousness instead of discussing the clearly pointed out reason for obviousness, it fails to pass close inspection for at least two reasons. First, this is actually an argument against **Klapproth** alone regardless of any combinations. Appellant isn't saying **Krishnaswamy** and **Klapproth** produce too many layers of confusing circuitry, Appellant's examples are to **Klapproth**'s circuitry layout itself. **Klapproth** describes the actual implementation of a working debugging system and circuitry, which by virtue of existence isn't non-obvious in view of itself. Second, Appellant's own claimed invention "the memory array being separate and distinct from the memory hierarchy" as viewed in figures 1 and 2, is "many layers and levels of circuit indirection".

C(iii)

Fifth in response to Appellant's C(iii), concerning applicant's argument that the examiner's conclusion of obviousness is based upon improper hindsight reasoning, it must be recognized that any judgment on obviousness is in a sense necessarily a reconstruction based upon hindsight reasoning. But

so long as it takes into account only knowledge which was within the level of ordinary skill at the time the claimed invention was made, and does not include knowledge gleaned only from the applicant's disclosure, such a reconstruction is proper. See *In re McLaughlin*, 443 F.2d 1392, 170 USPQ 209 (CCPA 1971).

In the particular case of scaling, **Computing** demonstrated the concept of scaling in order to ensure variables/results are within range of finite storage equipment (page 432, definition of scaling). **Krishnaswamy** demonstrated finite storage equipment (counters/registers) through the PMU (column 6, line 24). The PMU counts events and records them for a running/executing processor CPU (column 6, lines 21-30). Thus, it would be obvious to one of ordinary skill in the art to scale data for storage in a limited/finite equipment. Otherwise, the equipment would fail to accurately represent the counted events.

And Sixth in response to Appellant's E(iii), a properly cited motivation to combine **Krishnaswamy** and **Chang** was provided (see rejection of claims 41 and 42). Appellant has provided no analysis to contradict this motivation.

Finally, all additional arguments presented by Appellant are addressed by the above responses. Having addressed all of Appellant's arguments, the rejections are deemed proper.

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

William H. Wood

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Conferees:

Kakali Chaki

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

Tuan Dam

TUAN DAM
SUPERVISORY PATENT EXAMINER